

Detection and Visualization of Closed Streamlines in Planar Flows

Thomas Wischgoll and Gerik Scheuermann, *Member, IEEE*

Abstract—The analysis and visualization of flows is a central problem in visualization. Topology-based methods have gained increasing interest in recent years. This article describes a method for the detection of closed streamlines in flows. It is based on a special treatment of cases where a streamline reenters a cell to prevent infinite cycling during streamline calculation. The algorithm checks for possible exits of a loop of crossed edges and detects structurally stable closed streamlines. These global features are not detected by conventional topology and feature detection algorithms.

Index Terms—Vector field topology, limit cycles, closed streamlines.

1 INTRODUCTION

AN intuitive and often used method for vector field visualization is the calculation of streamlines. If one uses this technique in turbulent fields, one often encounters the problem of closed streamlines. A similar problem is discussed in [6], where residence time is used to find recirculation regions. When reaching a closed streamline, the residence time is infinite.

The difficulty with standard integration methods is that streamlines approaching a closed curve cycle around that curve without ever approaching a critical point or the boundary. Usually, one uses a stopping criteria like elapsed time or number of integration steps to prevent infinite loops. We present here an algorithm that detects this behavior and that can be used to visualize closed streamlines since these topological features are an essential topological property of the field.

Topological methods have received increasing interest in scientific visualization since their introduction by Helman and Hesselink [4], [7], [11], [14, chapter 21], [17], [16]. Our problem here is also related to the study of dynamical systems [5], [9], which have also been an application area for visualization. Koçak et al. [12] concentrate on the use of computer graphics for understanding Hamiltonian systems that appear frequently in mechanics. Hepting et al. [8] study invariant tori in four-dimensional dynamical systems by using suitable projections into three dimensions to enable detailed visual analysis of the tori. Wegenkittl et al. [20] present visualization techniques for known features of dynamical systems. Bürkle et al. [2] use a numerical algorithm developed by some of the coauthors [3] to visualize the behavior of more complicated dynamical systems. In the numerical literature, we can find several algorithms for the calculation of closed curves in dynamical

systems [10], [19], but these algorithms are tailored to deal with smooth dynamical systems where a closed form solution is given. In contrast, visualization faces, far more often, piecewise linear or bilinear vector fields. Here, the knowledge of the grid and the linear structure of the field in the cells allow a direct approach for the search of closed streamlines. The algorithm can be integrated in the streamline calculation as we will show. We repeat necessary terms on vector field topology in Section 2. Section 3 describes the algorithm for detecting closed streamlines. Results are presented in Section 5. Research questions are discussed, together with conclusions, in Section 6.

2 THEORY

This section repeats the theoretical background and the terms used in vector field topology which are used for our algorithm. The description of the Poincaré map mainly follows Abraham and Shaw [1].

2.1 Topology and Limit Cycles

The topology of planar vector fields concentrates on origin and end of streamlines. By combining curves of equal behavior into regions, the information contained in the field is concentrated. The boundaries of these regions are curves and points building a graph called *topological skeleton*. The origins or ends of streamlines are often critical points [7] or the boundary. Another common case appears if closed streamlines are present. This is often the case in plane vector fields on cutting planes because the third component is missing. This article shows how to detect and visualize these cycles.

We restrict our consideration in this article to planar, steady vector fields $v : \mathbb{R}^2 \supset D \rightarrow \mathbb{R}^2$, $(x, y) \mapsto v(x, y)$. D is assumed to be bounded. This is the situation for many experimental or simulated planar vector fields that have to be visualized. We are interested in the behavior of streamlines

$$c_a : \mathbb{R} \rightarrow \mathbb{R}^2, \quad t \mapsto c_a(t), \quad (1)$$

with the properties

• The authors are with the FB Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany.
E-mail: {wischgoll, scheuer}@informatik.uni-kl.de.

Manuscript received 4 Apr. 2000; revised 14 Sept. 2000; accepted 14 Sept. 2000.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number 111843.

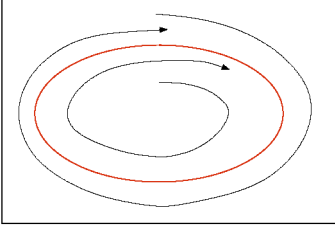


Fig. 1. A limit cycle may attract streamlines in its neighborhood.

$$c_a(0) = a \quad (2)$$

$$\frac{\partial c_a}{\partial t}(t) = v(c_a(t)). \quad (3)$$

For Lipschitz continuous vector fields, we can prove the existence and uniqueness of streamlines c_a through any point $a \in D$, see [9], [13]. The actual computation of the streamlines is usually done by numerical algorithms like Euler methods, Runge-Kutta-Fehlberg methods, or Predictor/Corrector methods [15], [18].

The topological analysis of vector fields considers the asymptotic behavior of streamlines. The origin set or α -limit set of a streamline c is defined by

$$\{p \in \mathbb{R}^2 \mid \exists (t_n)_{n=0}^\infty \subset \mathbb{R}, t_n \rightarrow -\infty, \lim_{n \rightarrow \infty} \alpha(t_n) \rightarrow p\}.$$

The end set or ω -limit set of a streamline α is defined by

$$\{p \in \mathbb{R}^2 \mid \exists (t_n)_{n=0}^\infty \subset \mathbb{R}, t_n \rightarrow \infty, \lim_{n \rightarrow \infty} \alpha(t_n) \rightarrow p\}.$$

If the α - or ω -limit set of a streamline consists of only one point, this point is a critical point or a point at the boundary ∂D of our domain D . (It is assumed that the streamline stays at the boundary point forever in this notation.) The critical points are divided into sources, sinks, and saddles. A *source* has a neighborhood where all streamlines start at the source. A *sink* has a neighborhood where all streamlines end at the sink. A *saddle* has only a finite number of streamlines that start or end at the saddle. All other streamlines in any neighborhood do not reach the saddle.

The most common case of an α - or ω -limit set in a planar vector field containing more than one inner point of the domain is a limit cycle [9]. This is a streamline c_a so that there is a $t_0 \in \mathbb{R}$ with

$$\alpha_a(t + nt_0) = \alpha_a(t) \quad \forall n \in \mathbb{N}. \quad (4)$$

Fig. 1 shows a typical example. Such a cycle is called structurally stable if, after small changes, the vector field still contains a closed streamline. Such stability arguments allow concentration on critical points, boundary regions, and closed cycles as possible α - and ω -limit sets [5], [9]. In this sense, this article completes the topological analysis of planar vector fields.

The union of all streamlines with the same origin is described by the α -basin

$$B_\alpha(S) = \{a \in D \mid S \text{ is the } \alpha\text{-limit set of } \alpha_a\},$$

where S may be a source, saddle, inflow set, or limit cycle. The union of all streamlines with the same end is described by the ω -basin $B_\omega(S) = \{a \in D \mid S \text{ is the } \omega\text{-limit set of } \alpha_a\}$. The final result of the topological analysis can be described as a decomposition of the domain D of the vector field in subsets with uniform behavior of the streamlines. To describe this, we denote with a_1, \dots, a_j the sources, with s_1, \dots, s_k the saddles, with z_1, \dots, z_l the sinks, and with b_1, \dots, b_m the boundary saddles of v . The inflow components of the boundary are denoted by I_1, \dots, I_n , the outflow components by O_1, \dots, O_o , and the boundary flow components by T_1, \dots, T_p . The attractive orbits are written as A_1, \dots, A_q and the repelling orbits by Z_1, \dots, Z_r . The domain $D \subset \mathbb{R}^2$ of the vector field can now be decomposed into α -basins

$$D = \bigcup_{i=1}^j B_\alpha(a_i) \cup \bigcup_{i=1}^n B_\alpha(I_i) \cup \bigcup_{i=1}^q B_\alpha(A_i) \cup \bigcup_{i=1}^k B_\alpha(s_i) \\ \cup \bigcup_{i=1}^m B_\alpha(b_i) \cup \bigcup_{i=1}^r Z_i \cup \bigcup_{i=1}^o O_i \cup \bigcup_{i=1}^l \{z_i\}$$

and into ω -basins

$$D = \bigcup_{i=1}^l B_\omega(z_i) \cup \bigcup_{i=1}^o B_\omega(O_i) \cup \bigcup_{i=1}^r B_\omega(Z_i) \cup \bigcup_{i=1}^k B_\omega(s_i) \\ \cup \bigcup_{i=1}^m B_\omega(b_i) \cup \bigcup_{i=1}^q A_i \cup \bigcup_{i=1}^n I_i \cup \bigcup_{i=1}^j \{a_i\}.$$

The topological analysis gives the connected components of the intersections between α and ω -basins. The visualization shows the boundaries of these components so that the components themselves become visible.

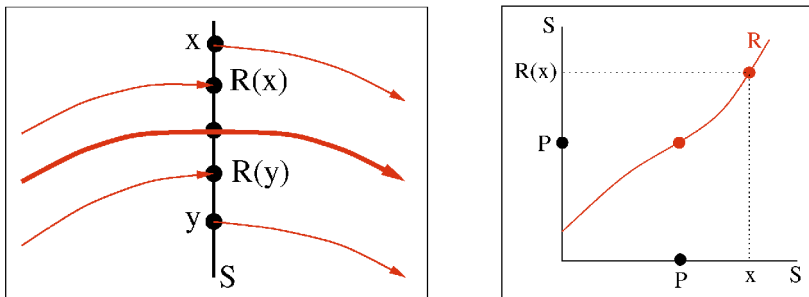


Fig. 2. Poincaré section and Poincaré map.

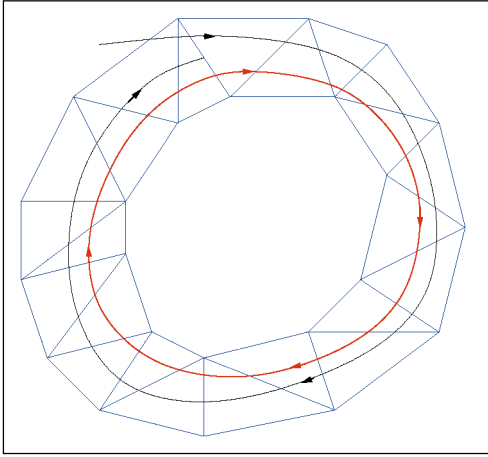


Fig. 3. A streamline approaching a limit cycle has to reenter cells.

2.2 Poincaré-Map

Let us assume we have a two-dimensional vector field containing one limit cycle. Then, we can choose a point P on the limit cycle and draw a *cross section* S which is a line segment nowhere parallel to the vector field. This line is then called a *Poincaré section*. If we start a streamline at an arbitrary point x on S and follow it until we cross the Poincaré section S again, we get another point $R(x)$ on S . This results in the *Poincaré map* R . Fig. 2 illustrates the situation. The left part shows the Poincaré section with the limit cycle in the middle drawn with a thicker line, while the right part displays the Poincaré map itself. Obviously, the point P on the limit cycle is a fix point of the Poincaré map.

3 DETECTION OF CLOSED STREAMLINES

As mentioned in Section 2.1, a closed streamline $\gamma: \mathbb{R} \rightarrow \mathbb{R}^2, t \mapsto \gamma(t)$ is a streamline of a vector field v such that there is a $t_0 \in \mathbb{R}$ with $\gamma(t + nt_0) = \gamma(t) \forall n \in \mathbb{N}$ and γ not constant. We present an algorithm that detects if an arbitrary streamline c converges to such a closed curve, also called a limit cycle. This means that c has γ as α or ω -limit set, depending on the orientation of integration. We do not assume any knowledge on the existence or location of the closed curve so that the algorithm can detect closed streamlines. The principle of the algorithm works on any piecewise defined planar vector field where one can determine the topology inside the pieces.

3.1 Finding Closed Streamlines

The basic idea of the algorithm is to determine a region of the vector field that is never left by the streamline. According to the Poincaré-Bendixson-Theorem, a streamline approaches a limit cycle if no singularity exists in that region. We assume that the data of the vector field is given on a grid consisting of triangles and/or quadrilaterals. The vectors inside a cell are interpolated so that we get at least a continuous vector field. In a precomputational step, we calculate every singularity of the vector field.

A streamline approaching a limit cycle has to reenter the same cell again, as shown in Fig. 3. In this case, we check if

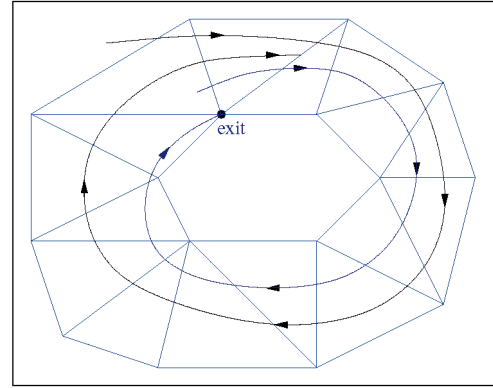


Fig. 4. If a real exit can be reached, the streamline will leave the cell cycle.

the cells crossed by the streamline have not changed for the last two turns. This results in a *cell cycle* which identifies the above-mentioned region. To examine if this cell cycle is left by the streamline, we detect possible changes by checking the edges of the cells of the cell cycle. Therefore, we find the points on each edge, which we call *potential exits*, where an outflow out of the cell cycle may occur near these points. These points are identical with the vertices of the edge and points where the vector field is tangential to the edge.

Then, we have to figure out if the actually investigated streamline will leave the cell cycle near such an exit. Therefore, we integrate a streamline backward from the potential exit to see if it leaves the cell cycle. If this is not the case, after the streamline crosses every cell involved in the cell cycle, it is shown that this backward integration converges to the streamline we actually investigate. Because the vectors at each edge are interpolated linearly, the streamline cannot turn around and cross the edge in the opposite direction between two vectors pointing in the direction of the streamlines, namely the two vectors we followed by the two turns of the actually investigated streamline while detecting the cell cycle. Therefore, it is sufficient to consider only one full turn of the backward integration, as in Fig. 4. We call this potential exit a *real exit* because the streamline will leave the cell cycle after a finite number of turns near that exit. Fig. 4 displays an example for that case.

If the backward integrated streamline leaves the cell cycle, i.e., it diverges from the actually investigated streamline, the streamline we want to check cannot leave the cell cycle in that potential exit because then we have an inflow into our region which will leave again at the exit, as shown in Fig. 5. Consequently, this is not a real exit.

It is sufficient to treat only the vertices of the cells and the points where the vector field is tangential because if the backward integration of any point on an edge leaves the cell cycle, the backward integration starting at one of these potential exits will also do. Fig. 6 shows the different configurations of potential exits. Let E be a point where we can leave the cell cycle. In Fig. 6a, both backward integrated streamlines starting at the vertices V_1 and V_2 leave the cell cycle. Consequently, E cannot be an exit because it has to cross one of the other backward integrated streamlines. Because of the linear interpolation at the edge, Fig. 6b is also

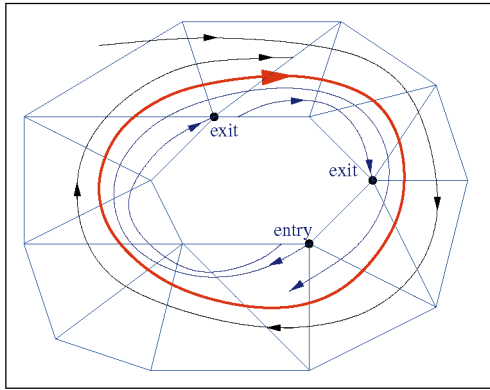


Fig. 5. If no real exit can be reached, the streamline will approach a limit cycle.

impossible. To get a possible configuration, the backward integration starting at the vertex V_1 also must converge the streamline because it cannot cross the backward integration starting at point E as in Fig. 6c. Fig. 6d explains why we also need to investigate the tangential case. If we start a backward integrated streamline at point E , it converges toward the streamline we actually investigate. But, if we only consider the vertices of the edge, both exits are not real exits. Therefore, we also have to start a backward integrated streamline at the point T , where the vector field is

tangential to the edge, to figure out that we leave the cell cycle at this edge.

If there is no real exit for the streamline, we have proven that the streamline will never leave the cell cycle. Because of the Poincaré-Bendixson-Theorem, there exists a closed streamline in our cell cycle and the integral curve tends toward it. If we can find a real exit, we have to continue the streamline calculation.

Fig. 7 illustrates the situation which shows a real example which is discussed in Section 5. There, we start a streamline near the source in the center of the figure. This streamline spirals until we find the first cell cycle, where we stopped the integration for this example. The figure also shows all exits and its backward integrations which are drawn in blue color and the streamline itself colored black. The grid is displayed in light blue. In this example, every potential exit is shown. We can see in this example that potential exits which are passed by a backward integrated streamline do not need to be investigated because if the backward integrated streamline leaves the cell cycle, the other one will also do so. Fig. 8 shows this in detail. There, the backward integrated streamline starting at *Exit 2* also has to leave the cell cycle because it cannot cross the backward integrated streamline starting at *Exit 1*. In the other case, where the backward integrated streamline started at *Exit 1* stays inside the cell cycle, we have to continue the actually investigated streamline, anyway.

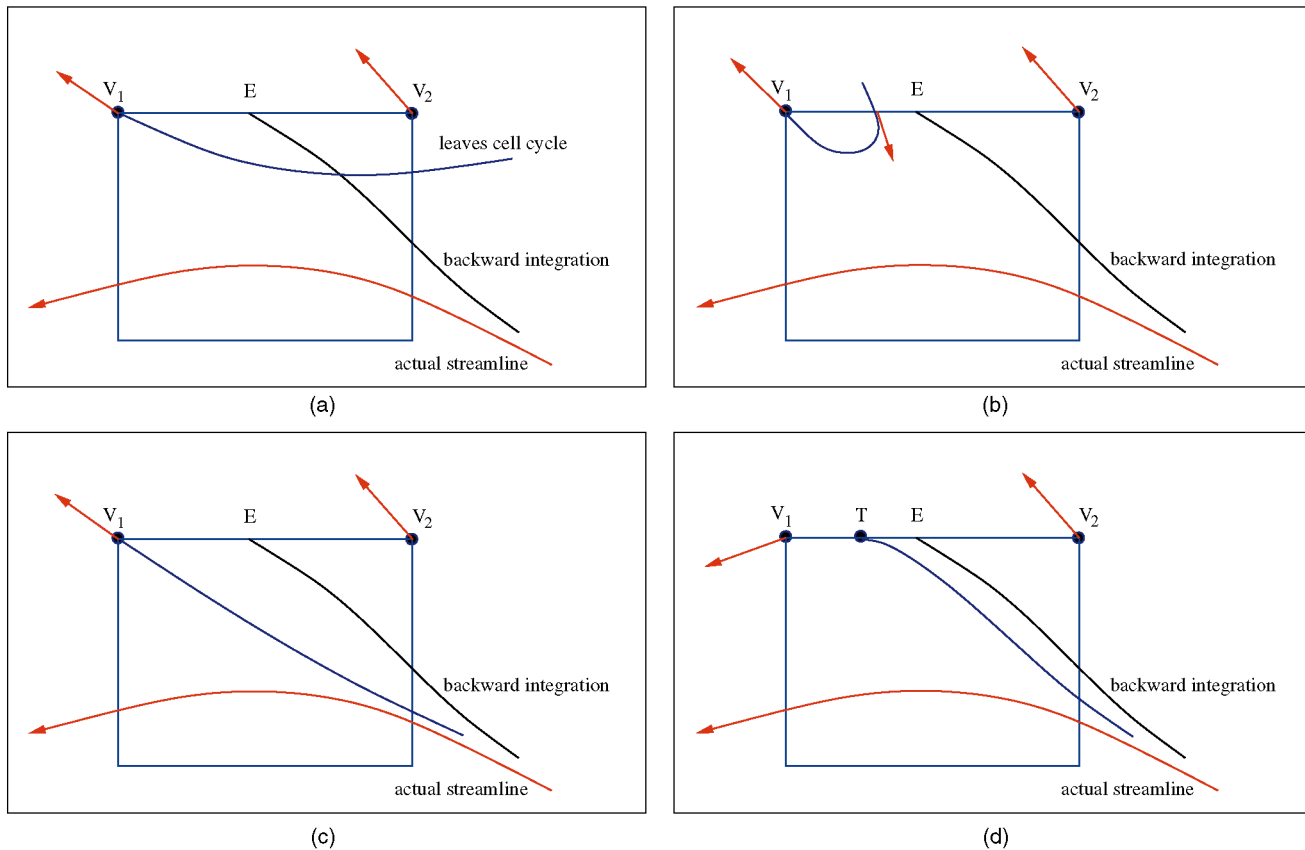


Fig. 6. Different cases of potential exits. (a) is impossible because streamlines cannot cross each other, (b) contradicts with the linear interpolation on an edge, in (c) and (d), both backward integrations converge toward the actual streamline so that the point E is a real exit.

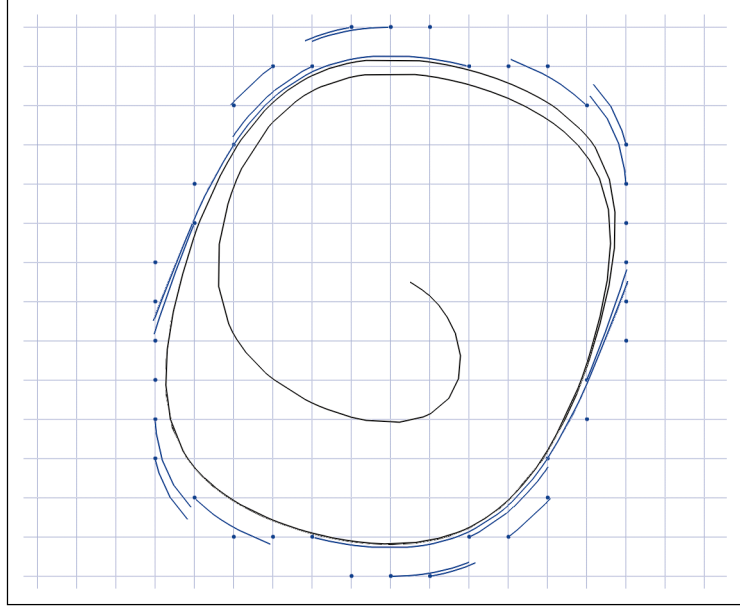


Fig. 7. Exits of a cell cycle.

Since the streamline spirals from the inner region to the outside, we only have to consider the potential exits in that direction. In the example, every backward integration leaves the cell cycle. Consequently, there is a limit cycle in this cell cycle which can be localized as described in Section 4.

In detail, our algorithm has three different states:

1. Streamline integration: Calculating one edge intersection after the other, check at each edge if we reached a cell cycle,
2. Checking for exit: Going backward through the intersected edges and looking for potential exit points,
3. Validating exit: Integrating backward a curve from the potential exit through all edges of the loop.

The algorithm switches its state after the events shown in Table 1, as can also be seen from the state diagram in Fig. 9.

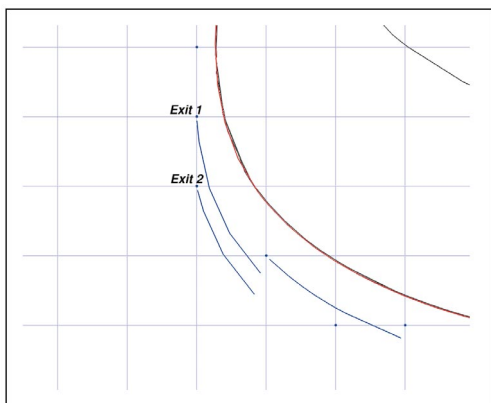


Fig. 8. Exits of the cell cycle which do not need to be investigated.

4 EXACT LOCATION OF THE CLOSED STREAMLINE

The exact position of the limit cycle can be found using the Poincaré map explained in Section 2.2, where we have to find the fix point to get a point on the limit cycle. If we find a cell cycle, we can use the edge where we detected the cell cycle for the first time as a Poincaré section. To find the fix point of the Poincaré map, we do a binary search: We divide the edge into two parts at the midpoint of the edge and check which part gets intersected by the streamline which is started at the intersection point after one turn. Then, this part is subdivided again and we start another streamline. This continues until we are close enough at the fix point of the Poincaré map. Then, we have determined one point of the limit cycle. If we start a streamline at this point, we get the whole limit cycle after we crossed every cell of the cell cycle. This method terminates because we proved in the previous step, where we detected the cell cycle, that we converge to a limit cycle.

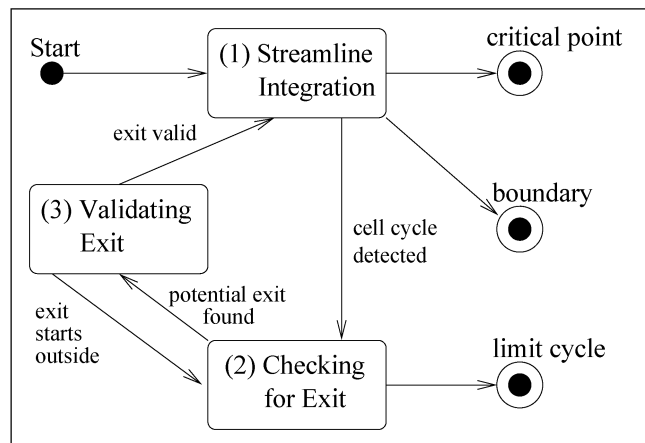


Fig. 9. The UML state diagram of our algorithm.

TABLE 1
Events and State Switching

Current state	new state	event
(1)	(2)	cell cycle detected
(2)	(3)	potential exit found
(3)	(2)	streamline from exit starts outside cell cycle
(3)	(1)	streamline from exit goes around the cell cycle
(1)	finished (boundary)	boundary edge reached
(1)	finished (crit. point)	critical point reached
(2)	finished (limit cycle)	first edge in cell cycle reached and no valid exit

5 RESULTS

Our first example is a vector field containing only one closed streamline. It was generated by sampling a slightly changed version of the *Van der Pol's equation* on a regular grid of sample points. The defining equation for the vector field V is

$$V \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y - x^3 + \mu x \\ -x \end{pmatrix}. \quad (5)$$

According to [9], a limit cycle is present for $0 < \mu \leq 1$ so that we can find a limit cycle. An analysis of the field shows that we have a source at $(0, 0)$. When starting our algorithm near that singularity, it integrates the streamline until it detects the limit cycle, as shown in Fig. 10. Fig. 10 also includes the hedgehog of the vector field, a glyph visualization method where we use arrows representing the vectors at the corresponding position. The arrows are twice as long as the vectors of the field.

In Fig. 11, we investigate a vector field which spirals from the singularity to the outer regions. Again, we used (5), but we set $\mu = -0.02$ to compute the vector field. Consequently, there is no limit cycle in the vector field and our algorithm correctly fails to detect one when started near the singularity at $(0, 0)$ and continues the streamline computation until the edge of the vector field is reached. Here, the hedgehog of the vector field is also displayed, scaled by factor of two.

The third example is a simulation of a swirling jet with an inflow into a steady medium. The simulation originally resulted in a three-dimensional vector field, but we used a cutting plane and projected the vectors onto this plane to get a two-dimensional field. In this application, one is interested in investigating the turbulence of the vector field and in regions where the fluid stays very long. This is necessary because some chemical reactions need a special amount of time. These regions can be located by finding closed streamlines. Fig. 12 shows the hedgehog of that vector field scaled by a factor of two. In Fig. 13, one can see some of the closed streamlines detected by our algorithm. All these limit cycles are located in the upper region of the vector field. Additionally, Fig. 13 includes the hedgehog, where the arrows representing the vectors are four times longer than the corresponding vector.

To compare our enhancements to the usual streamline computation methods, we implemented an algorithm which computes the topological skeleton as described in [7]. Therefore, we have to determine the singularities. Then, we start a streamline at each saddle point, displaced a little bit in the positive and negative eigendirection of both eigenvectors. Remember that our algorithm does not need any exit conditions other than the detection of closed streamlines or reaching a singularity or the border of the data!

To get an idea of the computational cost of our method, we also implemented a simple ODE solver to compute the

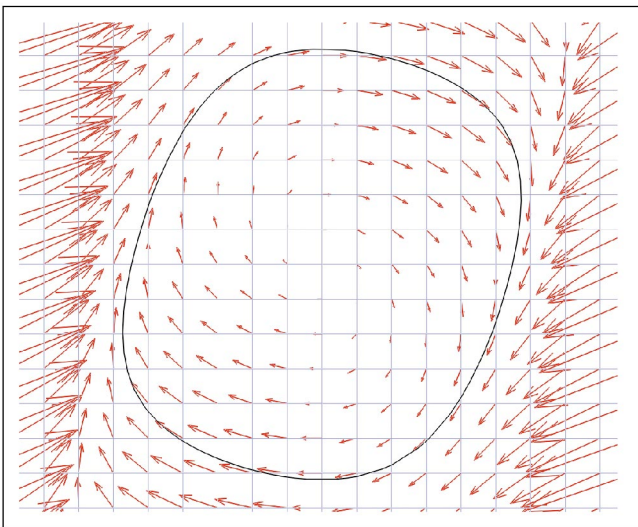


Fig. 10. Simple vector field with limit cycle.

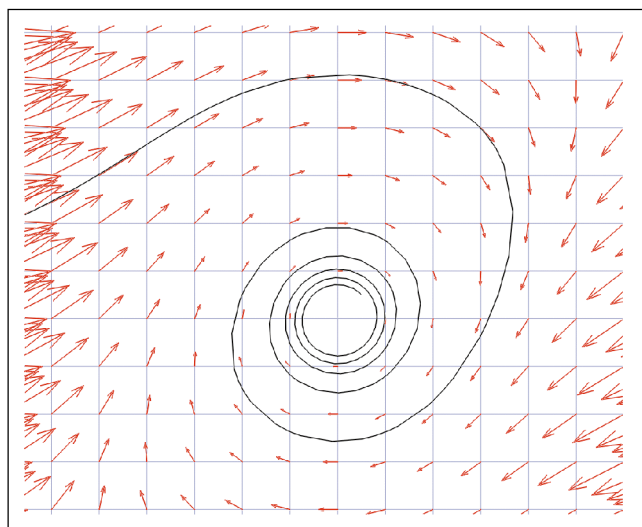


Fig. 11. Simple vector field with no limit cycle.

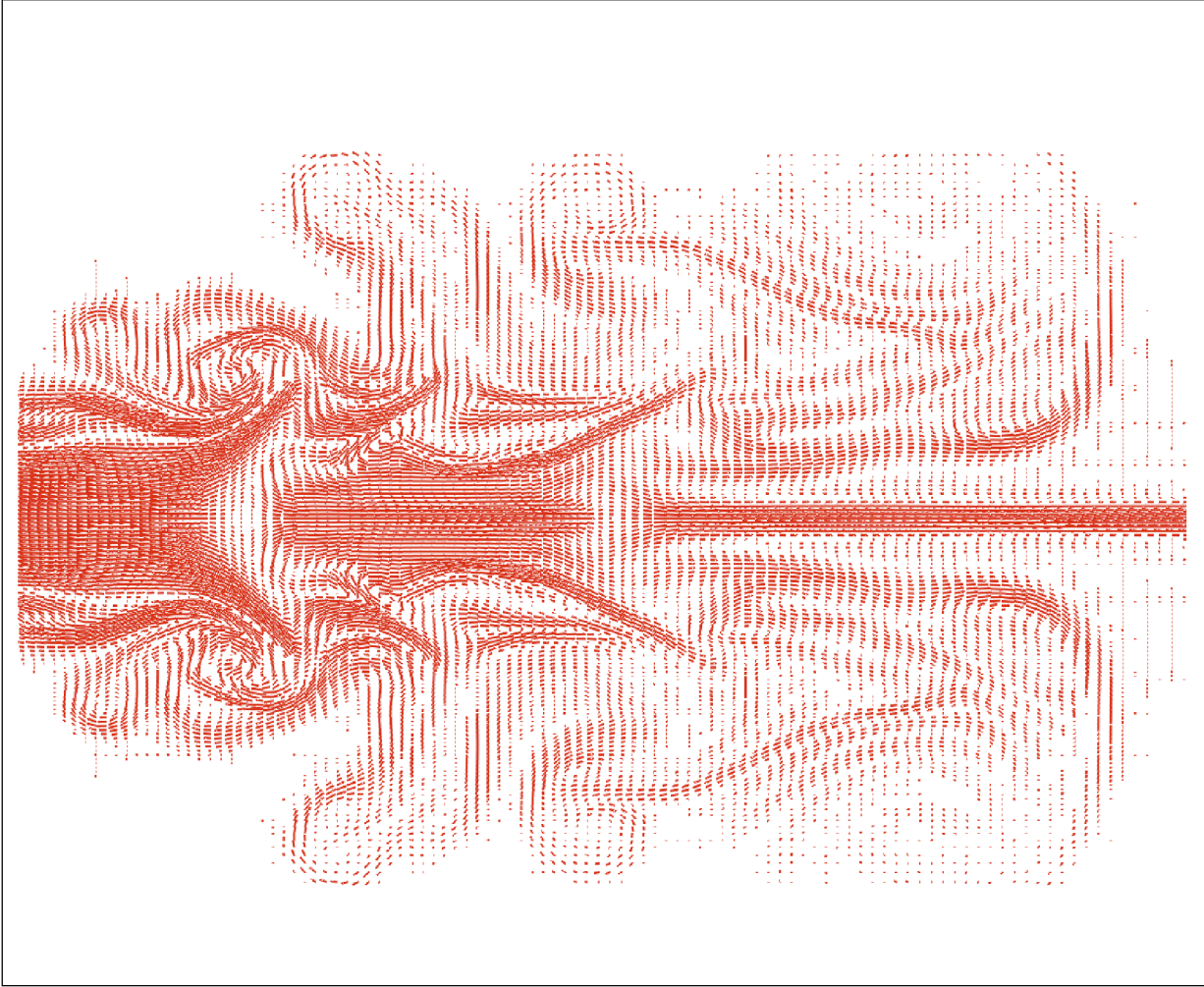


Fig. 12. Vorticity vector field of a turbulent flow—hedgehog.

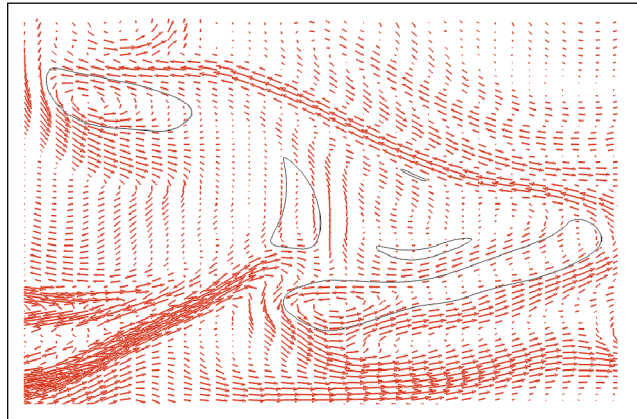
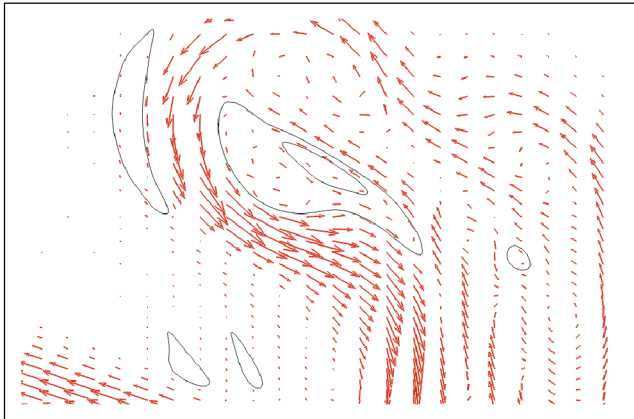


Fig. 13. Vorticity vector field of a turbulent flow—limit cycles.

streamlines. The vector field shown in Fig. 12 contains 337 singularities. The algorithm using a simple ODE solver needed 738 seconds to compute the topological skeleton on a Pentium II 350 MHz. Using our streamline integration method, which uses the same ODE solver but checks for limit cycles, we only needed 604 seconds on the same system, which is 18 percent faster! The reason for that is that we do not need to wait until the ODE solver reaches a

certain number of steps if we run into a limit cycle, which saves some time which we can use to check for limit cycles.

6 CONCLUSION

We have presented an algorithm for the calculation of streamlines that detects cycling around closed streamlines. Since it uses no information on the existence or location of

closed streamlines, it can be used to find these important topological features. The algorithm uses knowledge about the vector field which is interpolated linearly at the edge of the cells. We tested the algorithm with linear interpolation on a triangular grid and bilinear interpolation with quadrilateral cells. The algorithm theoretically should work on other cell types, too, but then it must be considered that streamlines may cross the edges in both directions, even in the cell cycle, which means that the streamline may turn around. The actual implementation sometimes has problems if there is more than one limit cycle crossing the same cell because the algorithm may not find the cell cycle or the exact position of only one limit cycle. This problem is the subject of current research of our group.

ACKNOWLEDGMENTS

This research was supported by the DFG project "Visualisierung nicht-linearer Vektorfeldtopologie." The authors would further like to thank Tom Bobach, Holger Burbach, Stefan Clauss, Jan Frey, Christoph Garth, Aragorn Rockstroh, René Schätzl, and Xavier Tricoche for their programming efforts. The continuing support of all members of the computer graphics and visualization team in Kaiserslautern gives us a nice working environment. Wolfgang Kollmann, MAE Department of the University of California at Davis, provided us with the vorticity dataset. We are very grateful for this and several helpful hints and discussions. Some ideas were developed during the postdoctoral work of Gerik Scheuermann at the CIPIC lab in Davis, so the authors would like to thank Bernd Hamann, Ken Joy, Kwan-Liu Ma, Nelson Max, and Jörg Meyer, as well as all other members of the group, for the fruitful cooperation.

REFERENCES

- [1] R.H. Abraham and C.D. Shaw, *Dynamics—The Geometry of Behaviour*, vol. 2. Santa Cruz, Calif.: Aerial Press, 1983.
- [2] D. Bürkle, M. Dellnitz, O. Junge, M. Rumpf, and M. Spielberg, "Visualizing Complicated Dynamics," *Proc. IEEE Visualization '99 Late Breaking Hot Topics*, A. Varshney, C.M. Wittenbrink, and H. Hagen, eds., pp. 33-36, 1999.
- [3] M. Dellnitz and O. Junge, "On the Approximation of Complicated Dynamical Behavior," *SIAM J. Numerical Analysis*, vol. 36, no. 2, pp. 491-515, 1999.
- [4] A. Globus, C. Levit, and T. Lasinski, "A Tool for Visualizing the Topology of Three-Dimensional Vector Fields," *Proc. IEEE Visualization '91*, G.M. Nielson and L. Rosenblum, eds., pp. 33-40, 1991.
- [5] J. Guckenheimer and P. Holmes, *Dynamical Systems and Bifurcation of Vector Fields*. New York: Springer, 1983.
- [6] R. Haimes, "Using Residence Time for the Extraction of Recirculation Regions," *AIAA Paper 99-3291*, 1999.
- [7] J.L. Helman and L. Hesselink, "Visualizing Vector Field Topology in Fluid Flows," *IEEE Computer Graphics and Applications*, vol. 11, no. 3, pp. 36-46, May 1991.
- [8] D.H. Hepting, G. Derks, D. Edoh, and R.R. D., "Qualitative Analysis of Invariant Tori in a Dynamical System," *Proc. IEEE Visualization '95*, G.M. Nielson and D. Silver, eds., pp. 342-345, 1995.
- [9] M.W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra*. New York: Academic Press, 1974.
- [10] M. Jean, "Sur la Méthode des Sections pour la Recherche de Certaines Solutions presque Périodiques de Systèmes Forces Periodiquement," *Int'l J. Non-Linear Mechanics*, vol. 15, pp. 367-376, 1980.
- [11] D.N. Kenwright, "Automatic Detection of Open and Closed Separation and Attachment Lines," *Proc. IEEE Visualization '98*, pp. 151-158, D. Ebert, H. Rushmeier, and H. Hagen, eds., 1998.
- [12] H. Koçak, F. Bisshop, T. Banchoff, and D. Laidlaw, "Topology and Mechanics with Computer Graphics," *Advances in Applied Math.*, vol. 7, pp. 282-308, 1986.
- [13] S. Lang, *Differential and Riemannian Manifolds*, third ed. New York: Springer, 1995.
- [14] *Scientific Visualization, Overviews, Methodologies, and Techniques*, G.M. Nielson, H. Hagen, and H. Müller, eds. Los Alamitos, Calif.: IEEE CS Press, 1997.
- [15] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
- [16] G. Scheuermann, B. Hamann, K.I. Joy, and W. Kollmann, "Visualizing Local Vector Field Topology," *J. Electronic Imaging*, vol. 9, no. 4, 2000.
- [17] G. Scheuermann, H. Krüger, M. Menzel, and A. Rockwood, "Visualizing Nonlinear Vector Field Topology," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 2, pp. 109-116, Apr.-June 1998.
- [18] J. Stoer and R. Bulirsch, *Numerische Mathematik 2*, third ed. Berlin: Springer, 1990.
- [19] M. van Veldhuizen, "A New Algorithm for the Numerical Approximation of an Invariant Curve," *SIAM J. Scientific and Statistical Computing*, vol. 8, no. 6, pp. 951-962, 1987.
- [20] R. Wegenkittl, H. Löffelmann, and E. Gröller, "Visualizing the Behavior of Higher Dimensional Dynamical Systems," *Proc. IEEE Visualization '97*, R. Yagel and H. Hagen, eds., pp. 119-125, 1997.



Thomas Wischgoll received his Master's degree in computer science in 1998 from the University of Kaiserslautern. His master's thesis concerned the field of information visualization resulting in the visualization of temporal distances. Currently, he is working as a research assistant in the Computer Science Department of the University of Kaiserslautern. His research interests include vector field topology, parallel computing, and scientific visualization.



Gerik Scheuermann received the BS and MS degrees in mathematics in 1995 from the University of Kaiserslautern. In 1999, he received a PhD degree in computer science, also from the University of Kaiserslautern. During 1995-1997, he conducted research at Arizona State University for about a year. He has worked as a postdoctoral researcher at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California at Davis. Currently, he is working as a senior research scientist in the Computer Science Department of the University of Kaiserslautern. His research topics include algebraic geometry, topology, Clifford algebra, and scientific visualization. Dr. Scheuermann is a member of the IEEE.

► For further information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.